Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Jan Sequens

RDFa editor

Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Martin Nečaský
Studijní program: Informatika, obor Programování

2008

Děkuji panu Mgr. Martinu Nečaskému za odborné vedení mé práce a čas, který mi během jejího vzniku věnoval. Dále bych rád poděkoval rodině a přátelům, kteří mě při tvorbě práce podporovali.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 7. 8. 2008 Jan Sequens

Obsah

Název práce: RDFa editor
Autor: Jan Sequens
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí bakalářské práce: Mgr. Martin Nečaský
e-mail vedoucího: martinnec@gmail.com

Abstrakt: Jazyk RDFa je jednoduchým rozšířením jazyka XHTML pro tzv. sémantickou anotaci XHTML dokumentů. Možnosti samotného jazyka XHTML pro popis sémantiky dat jsou velice omezené. Tento problém se snaží vyřešit právě jazyk RDFa spojením XHTML a technologií sémantického webu. Sémantika je popsána pomocí ontologie a XHTML dokumenty jsou pomocí RDFa rozšíření navázány na koncepty v této ontologii. Cílem práce bylo vytvořit editor, který by RDFa anotaci usnadňoval. Kromě přímé anotace s automatickou kontextovou nápovědou atributů umožňuje editor import ontologií, vkládání jejich prvků metodou drag&drop a pro kontrolu také zobrazování vygenerovaných RDF trojic.

Klíčová slova: RDFa, XHTML, W3C, sémantický web

Title: RDFa editor
Author: Jan Sequens
Department: Department of Software Engineering
Supervisor: Mgr. Martin Nečaský
Supervisor's e-mail address: martinnec@gmail.com

Abstract: RDFa is a set of extensions to XHTML which allows to annotate XHTML markup with semantics. Because XHTML does not offer much semantic annotation potential itself, the RDFa language was proposed to relieve this shortage by combining XHTML and semantic web technologies. While semantics is being described by ontologies, the XHTML documents are linked to ontologies concepts by RDFa extensions. In the present work I tried to create and describe an editor that could make RDFa annotation easier. In addition to direct annotation with context attributes hints it provides ontology imports with drag&drop items insertion and extracted RDF triples preview.

Keywords: RDFa, XHTML, W3C, semantic web

Kapitola 1

Úvod

Internet se stává nedílnou součástí života čím dál tím většího počtu lidí. Množství informací, které ukrývá, dramaticky roste a očekává se, že bude růst i nadále. Tento trend s sebou přináší vzrůstající nároky na strojové vyhledávání a další zpracování těchto informací.

V současné době je nejběžnějším způsobem vyhledávání fulltextové. To je založené na porovnávání uživatelem zadaných znakových řetězců s řetězci získanými z vyhledávacím strojem indexovaných stránek. Při tomto způsobu vyhledávání není nijak zohledněn význam hledaných řetězců, což mnohdy činí nalezení relevantních informací obtížným.

Řešením tohoto problému se má stát idea tzv. sémantického webu, navržená konsorciem W3C, jejíž součástí je i jazyk RDFa.

RDFa je sadou rozšíření jazyka XHTML umožňující provázat obsah internetových stránek s entitami z tzv. ontologií, které bychom mohli nepřesně označit jako slovníky pojmů. Kromě samotných pojmů ontologie ale definují zejména vztahy mezi těmito pojmy.

Myšlenka sémantického webu je tedy založena na představě, že webové stránky budou opatřeny strojově čitelnými poznámkami, které umožní data automaticky zpracovávat, třídit a hledat s přihlédnutím k jejich skutečnému významu v reálném světě.

Navzdory velikosti a významu tohoto konceptu se zatím RDFa používá jen ve značně omezené komunitě a jeho masivní šíření se stále příliš nedaří.

Cílem této práce je vytvořit a popsat editor, který by sémantickou anotaci webových stránek pomocí RDFa zjednodušil a zpřístupnil širšímu okruhu uživatelů.

Kapitola 2

Teorie

2.1 Sémantický web

Sémantický web, označovaný také někdy jako Web 3.0, je dalším očekávaným evolučním stupněm Internetu. Jeho myšlenku, která je nejen technického, ale i filosofického rázu, vyslovil poprvé sir Timothy Berners-Lee, vynálezce World Wide Webu a ředitel konsorcia W3C (World Wide Web Consortium) v roce 2001.

Berners-Lee upozornil na skutečnost, že web se stává nepřehledným skladištěm informací, přičemž metody pro jejich vyhledávání a třídění přestávají být dostačující. V rámci svého působení ve W3C popsal sémantický web jako web, který umožňuje efektivní komunikaci mezi lidmi a počítači, výměnu informací mezi aplikacemi podle standardizovaných pravidel, integraci dat získaných z různých zdrojů a popis vztahů mezi daty a reálnými objekty.

Kromě zmíněného filosofického aspektu se sémantickým webem rozumí soubor technických specifikací, mezi které patří zejména Resource Description Framework (RDF), RDF Schema (RDFS), Web Ontology Language (OWL) a SPARQL Protocol and RDF Query Language (SPARQL), jejichž účelem je poskytnout prostředky pro formální popis entit reálného světa a vztahů mezi nimi.

Přestože Berners-Lee propaguje sémantický web již řadu let, nedočkal se zatím výraznějšího rozmachu a masového přijetí. Jednou z pravděpodobných příčin je přílišná komplikovanost ve srovnání se stávajícím webem. Mezi další kritické reakce na koncept sémantického webu patří zmínky o možnosti snadné cenzury, potlačení záměrné anonymity bloggerů a zbytečném zdvo-

jování obsahu, kdy by se každá stránka musela vytvářet ve verzi pro uživatele a pro strojové zpracování.

Zajímavou snahou o jednoduchý způsob sémantické anotace dokumentů jsou tzv. mikroformáty (microformats), které ovšem nepochází z dílny W3C (ale nejsou v rozporu s žádným z jeho doporučení); jejich autorem je Tantek Çelik. Mikroformáty se pokouší k anotaci využít existujících HTML elementů a atributů, konkrétně tvůrcům webových stránek dobře známého atributu `class`. Souvisejícím formátem je Gleaning Resource Descriptions from Dialects of Languages (GRDDL), který umožňuje z mikroformátů získat data ve formátu RDF.

Dalším projektem, který si klade za cíl zjednodušit sémantickou anotaci webových dokumentů a který byl již vytvořen v rámci W3C, je jazyk RDFa.

2.2 Resource Description Framework

Resource Description Framework (RDF) je rodina W3C specifikací navržených jako model metadat (informací o informacích). RDF sám o sobě neurčuje konkrétní syntaxi těchto metadat, je pouze jakousi výchozí abstrakcí.

Základním stavebním kamenem je deklarace ve formě trojice subjekt-predikát-objekt, označovaná jednoduše jako *trojice* (anglicky *triple*). Predikát určuje vztah mezi subjektem a objektem. RDF deklarace mají tedy zjevnou analogii v konstrukci vět v obyčejném jazyce a jsou chápany vcelku intuitivně.

Subjekt i predikát jsou vždy představovány *zdrojem* (*resource*), zatímco objekt může být představován zdrojem nebo datovou hodnotou.

Zdroje jsou vyjádřeny pomocí URI (Uniform Resource Identifiers) neboli jedinečných identifikátorů zdrojů – znakových řetězců s definovanou strukturou. URI může být klasifikován jako URL (Uniform Resource Locator), označující umístění zdroje, nebo URN (Uniform Resource Name), označující jeho unikátní jméno.

Pro zápis RDF existuje několik syntaxí. Jako první byla představena XML (Extensible Markup Language) syntaxe, označovaná jako RDF/XML. Její nevýhodou je však přílišná obsáhlost, která může činit „ruční“ zápis nepohodlným. Proto byla později představena syntaxe Notation 3 (či zkráceně N3), která není založena na XML, a RDF trojice jsou v ní snáze rozpoznatelné. Podmnožinou Notation 3 je pak syntaxe Turtle (Terse RDF Triple Language). Nejméně komplikovanou syntaxí jsou N-triples, což je prostý výčet RDF trojic.

Na závěr této kapitoly si ukažme krátký příklad [2] RDF/XML:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

<contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
  <contact:fullName>Eric Miller</contact:fullName>
  <contact:mailbox rdf:resource="mailto:em@w3.org"/>
  <contact:personalTitle>Dr.</contact:personalTitle>
</contact:Person>

</rdf:RDF>
```

V kořenovém elementu `rdf:RDF` definujeme dva jmenné prostory `rdf` a `contact`, přičemž první z nich (`rdf`) používáme dokonce i v samotném elementu, ve kterém byl tento prostor definován.

Dále říkáme, že URI `http://www.w3.org/People/EM/contact#me` je instancí třídy `Person`, a určíme hodnoty jejích vlastností – hodnotou vlastnosti `fullName` je datová hodnota `Eric Miller`, hodnotou vlastnosti `mailbox` je zdroj `mailto:em@w3.org` a hodnotou vlastnosti `personalTitle` je datová hodnota `Dr.`.

2.3 Ontologie

Pojem ontologie pochází z filosofie, kde označuje disciplínu zabývající se základními pojmy, jsoucnem a bytím jako takovým.

V souvislosti s informačními technologiemi je ontologie formální reprezentací pojmů z určité problematiky a vztahů mezi těmito pojmy. Přesných definic ontologie vzniklo několik, pro zajímavost můžeme uvést definici T. Grubera, jednoho z „duchovních otců“ ontologií: „ontologie je explicitní specifikace konceptualizace“ [6].

Souboru pojmů se někdy říká *glosář* a souboru vztahů *tezaurus*. Ontologie nalézají využití nejen v oblasti sémantického webu, ale také v umělé inteligenci, softwarovém inženýrství, medicínské informatice atd.

Základními prvky ontologií jsou

- *individua* – instance objektů, konkrétní objekty,
- *třídy* – typy či množiny objektů,
- *atributy* – vlastnosti či parametry objektů,
- *relace* – vazby mezi třídami či jedinci, je možné považovat za speciální případ atributů, jejichž hodnotami jsou jiní jedinci či třídy.

Pro popis ontologií jsou používány speciální jazyky, v případě použití ontologií v souvislosti se sémantickým webem se jedná zejména o Web Ontology Language (OWL).

2.4 Web Ontology Language

Web Ontology Language (OWL) je rodinou jazyků pro vytváření ontologií. Vznikal v letech 2002 až 2004 na základě staršího jazyka DAML+OIL a v roce 2004 se zařadil mezi oficiální doporučení konsorcia W3C.

OWL je založen na RDF a pro jeho zápis se nejčastěji používá RDF/XML syntaxe.

OWL existuje ve třech dialektech:

- *OWL Lite*,
- *OWL DL*,
- *OWL Full*.

Ty se liší zejména množstvím toho, co vše jimi lze vyjádřit, a zárukami výpočetní složitosti. Lze říci, že nejkomplexnější dialekt OWL Full je syntaktickým rozšířením OWL DL a OWL DL je syntaktickým rozšířením OWL Lite.

2.5 Jazyk RDFa

RDFa neboli Resource Description Framework attributes je sadou rozšíření jazyka XHTML (Extensible Hypertext Markup Language), která byla navržena v roce 2004 Markem Birbeckem v rámci činnosti konsorcia W3C.

Základem RDFa je soubor atributů, které umožňují přidat metadata do XHTML dokumentů. Z těchto metadat mohou být poté získány RDF trojice.

Množina atributů, které RDFa přidává do XHTML, je následující [3]:

- *about* – říká, k čemu se data vztahují (definuje subjekt v RDF trojici),
- *datatype* – vyjadřuje datový typ znakového řetězce,
- *property* – vyjadřuje jeden nebo více vztahů mezi subjektem a znakovým řetězcem (definuje predikát v jedné nebo více RDF trojicích),
- *resource* – vyjadřuje RDF „zdroj“ (definuje objekt v RDF trojici),
- *typeof* – určuje, se kterým datovým typem se má asociovat RDF subjekt.

Následující atributy jsou součástí XHTML, ale RDFa je využívá [3]:

- *content* – znakový řetězec reprezentující strojově čitelný ekvivalent jiného znakového řetězce (obsah atributu `content` definuje objekt v RDF trojici),
- *href* – definuje objekt v RDF trojici,
- *rel* – vyjadřuje jeden či více vztahů mezi RDF „zdroji“ (definuje predikát v jedné nebo více RDF trojicích),
- *rev* – stejné použití jako `rel`, ale vyjadřuje vztah v opačném směru a způsobí tak vzájemnou výměnu subjektu a objektu v relevantních RDF trojicích
- *src* – definuje objekt v RDF trojici.

Jelikož URI, kterými se RDF zdroje určují, bývají často dost dlouhé, umožňuje RDFa zápis pomocí XML jmenných prostorů. Ty umožňují definovat v elementech prefixy, platící ve všech dceřinných elementech. Tyto prefixy pak zápis RDF zdrojů značně zpřehledňují.

Použití několika RDFa atributů spolu s definicí prefixů ilustruje následující příklad:

```
...
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xml:lang="cs" version="XHTML+RDFa 1.0">
...
<span property="cal:dtstart" content="19841210T1700-0500"
      datatype="xsd:datetime">
  10. prosince 1984 v 17:00
</span>
...
```

Po zavedení zkratek `cal` a `xsd` pomocí atributu `property` říkáme, že obsah tohoto elementu je začátkem určité události. Jelikož však datum je tak, jak je obsaženo uvnitř našeho elementu `span`, špatně strojově čitelné, poskytneme jeho alternativní verzi v atributu `content` a atributem `datatype` deklarujeme, že se jedná o datový typ reprezentující datum a čas.

Výše popsané souvislosti mezi RDFa/XHTML atributy a generováním prvků RDF trojic jsou sice pravdivé, leč neúplné. Součástí vyhodnocení RDFa zápisu je totiž *řetězení* (*chaining*) neboli možnost ovlivňovat výsledné RDF trojice vnořováním XHTML elementů. Jednu z takových situací představuje následující příklad [3]:

```
<div about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name">Albert Einstein</span>
  <div rel="dbp:birthPlace"
       resource="http://dbpedia.org/resource/Germany">
    <span property="dbp:conventionalLongName">
      Federal Republic of Germany
    </span>
  </div></div>
```

V tomto případě je hodnota atributu `resource` nejen objektem ve vztahu „Albert Einstein – má místo narození – Německo“, ale i subjektem ve vztahu „Německo – má formální jméno – Spolková republika Německo“.

Kapitola 3

Analýza problematiky

V této kapitole se zamyslíme nad teoretickými možnostmi usnadnění sémantické anotace XHTML dokumentů. Budeme se také zabývat programy, které tyto možnosti implementují, a vymezíme, jakými funkcemi se bude zabývat tato práce.

3.1 Možná funkcionalita

Základním stavebním kamenem následujících úvah nechť je myšlenka, že pro sémantickou anotaci potřebujeme především jednoduchý textový editor. Textovým editorem rozumíme samozřejmě editor prostého, neformátovaného textu, nikoliv textový procesor.

Možnostmi usnadnění, popsanými v následujícím textu, tedy popisujeme funkcionalitu, která je nad rámec zmíněného editoru, a kterou by mohl vyspělý editor jazyka RDFa nabídnout.

3.1.1 Integrace FTP klienta

Kromě standardní práce se soubory, která obnáší otevírání, ukládání a vytváření souborů na lokálních discích, by bylo možné nabízet práci se soubory na vzdáleném FTP serveru. Za předpokladu stálého připojení k internetu by uživatel nepocítil mezi prací s lokálním a vzdáleným souborem žádné podstatné rozdíly.

Vzhledem k tomu, že program je zaměřen primárně na editaci XHTML dokumentů, tedy webových stránek, jistě by tato funkce nemalou měrou

přispěla k jeho praktičnosti. Na druhou stranu se nejedná o funkcionalitu specifickou pro RDFa.

3.1.2 Zvýrazňování syntaxe

Zvýrazňování syntaxe významně zvyšuje přehlednost zobrazovaného kódu, uživatel se rychleji orientuje a jeho práce je efektivnější.

Kromě zvýrazňování XML elementů, atributů a jejich hodnot by bylo vhodné zvláštní barvou zvýrazňovat ty atributy, které jsou využívány jazykem RDFa.

3.1.3 Napovídání atributů

Často opakovanou činností uživatele bude jistě psaní názvů RDFa atributů, což je činnost relativně časově náročná a náchylná ke zbytečným překlepům. Pro její usnadnění by program mohl během psaní názvu atributu automaticky nabízet jeho doplnění. Pokud by nebylo jednoznačné, který atribut uživatel začíná psát, bylo by nabídnuto atributů více. Uživatel by si kurzorovými šipkami vybral z nabídky a klávesou Enter potvrdil doplnění.

Kromě této funkcionality by program mohl automaticky nabízet vložení dalších atributů podle povahy atributů již přítomných. Příkladem může být nabídka vložení atributu `datatype`, pokud je přítomen atribut `content`, neboť specifikace datového typu pro hodnotu atributu `content` může být poměrně častým úkonem.

3.1.4 Zobrazování RDF

Jelikož jazyk RDFa je nástrojem pro zakomponování jazyka RDF do XHTML dokumentů, mohlo by být pro uživatele přínosné prohlédnout si samotné RDF, které jeho RDFa anotace generuje. To by pak mohlo být exportováno a dále využito.

Program by mohl nabízet několik formátů exportu RDF (N Triples, RDF/XML, Turtle atd.).

Kromě jednorázového zobrazení RDF by bylo možné nabízet automaticky obnovovaný náhled.

V případě zobrazení v syntaxi „N triples“ by mohlo být možné zobrazení filtrovat, např. zobrazovat pouze trojice obsahující určité URI apod.

3.1.5 Import ontologií

Příjemnou funkcí by byl jistě i import ontologií do programu, kdy by se z ontologie získaly názvy tříd a vlastností, které by mohly být dále využity, například pro přetažení do dokumentu metodou „drag&drop“.

Import ontologie by mohl nabídnout přidání jejího jmenného prostoru do dokumentu.

Skutečně pokročilý RDFa editor by mohl komunikovat s některou z na internetu operujících knihoven ontologií.

3.1.6 Napovídání pojmů z ontologií

Způsobem analogickým ke způsobu popsanému v 3.1.3 by mohly být uživateli v okamžiku psaní hodnoty RDFa atributu nabízeny pojmy získané z importovaných ontologií.

Rozšířením takové nápovědy by pak bylo protežování určitých pojmů jejich dočasným předřazením před pojmy ostatní, prováděné podle kontextu, ve kterém se právě anotovaný element nachází.

Jako příklad může posloužit situace, kdy uživatel provádí anotaci elementu, který je přímým potomkem elementu s atributem `typeof`. V takovém případě by bylo jistě vhodné nabízet přednostně ty pojmy, které jsou vlastnostmi třídy uvedené v hodnotě zmíněného atributu `typeof` v rodičovském elementu.

Napovídání pojmů z ontologií by mohlo zobrazovat pouze ty pojmy, které se nalézají ve jmenných prostorech, pro které jsou v dokumenty definovány prefixy. Opačným přístupem by bylo zobrazování všech dostupných pojmů a nabídka automatické definice vhodných prefixů při vložení pojmu.

3.1.7 Ostatní funkce

Zajímavou funkcí by jistě byla volba kontextového menu, jež by umožňovala snadné kopírování RDFa atributů (včetně jejich hodnot) z jednoho elementu do druhého.

Speciální menu by také mohlo umožňovat přímé vkládání názvů atributů, případně jiných znakových řetězců, které jsou pro tuto oblast charakteristické. Příkladem může být deklarace typu dokumentu XHTML+RDFa.

Rovněž by mohl být přítomen nástroj pro automatické odstranění veškeré RDFa anotace z dokumentu.

Pokročilou funkcí by pak mohla být automatická kontrola dodržení domény, oboru hodnot, kardinality a dalších omezení, která jsou ontologií na entity kladena.

3.2 Existující programy

Přestože například nástrojů pro převod RDFa na RDF je k dispozici vcelku velké množství, editor, který by byl použitelný pro RDFa anotaci XHTML dokumentů a implementoval by významnější množství výše popsaných funkcí, se mi navzdory dlouhému hledání podařilo objevit pouze jeden.

3.2.1 TopBraid Composer

TopBraid Composer [5] od společnosti TopQuadrant je komerčním vývojovým prostředím pro modelování ontologií. Je založen na platformě Eclipse. Koncem roku 2006 byl TopBraid Composer rozšířen o podporu práce s RDFa.

Podpora práce s RDFa je ztělesněna zejména těmito funkcemi:

- zvýrazňování syntaxe včetně vyznačení klíčových slov RDFa,
- napovídání atributů,
- zobrazování RDF trojic,
- přetahování pojmů z otevřených ontologií metodou „drag&drop“,
- napovídání pojmů z ontologií.

Z výčtu funkcí plyne, že TopBraid Composer je pro RDFa anotaci dokumentů – přestože to není jeho hlavním účelem – poměrně silným nástrojem. Disponuje příjemným grafickým uživatelským rozhraním, které je ovšem z pohledu uživatele, který se nezabývá modelováním ontologií, poněkud složitější, a jeví se tak být přes jeho hezké zpracování ne zcela přehledným.

3.3 Rozsah této práce z hlediska funkčnosti

Předmětem této práce bude implementace následující podmnožiny funkcí popsaných v 3.1:

- zvýrazňování syntaxe včetně vyznačení klíčových slov RDFa,
- jednoduché napovídání atributů,
- zobrazování RDF trojic ve formátu „N triples“,
- jednoduchý import ontologií,
- přetahování pojmů z ontologií metodou „drag&drop“.

Kapitola 4

Uživatelská dokumentace

4.1 Účel programu

Program RDFa editor je jednoduchým textovým editorem, který byl vytvořen za účelem usnadnění anotace XHTML souborů jazykem RDFa. Jeho funkčnost se tedy soustředí na specifické potřeby takové anotace.

4.2 Instalace

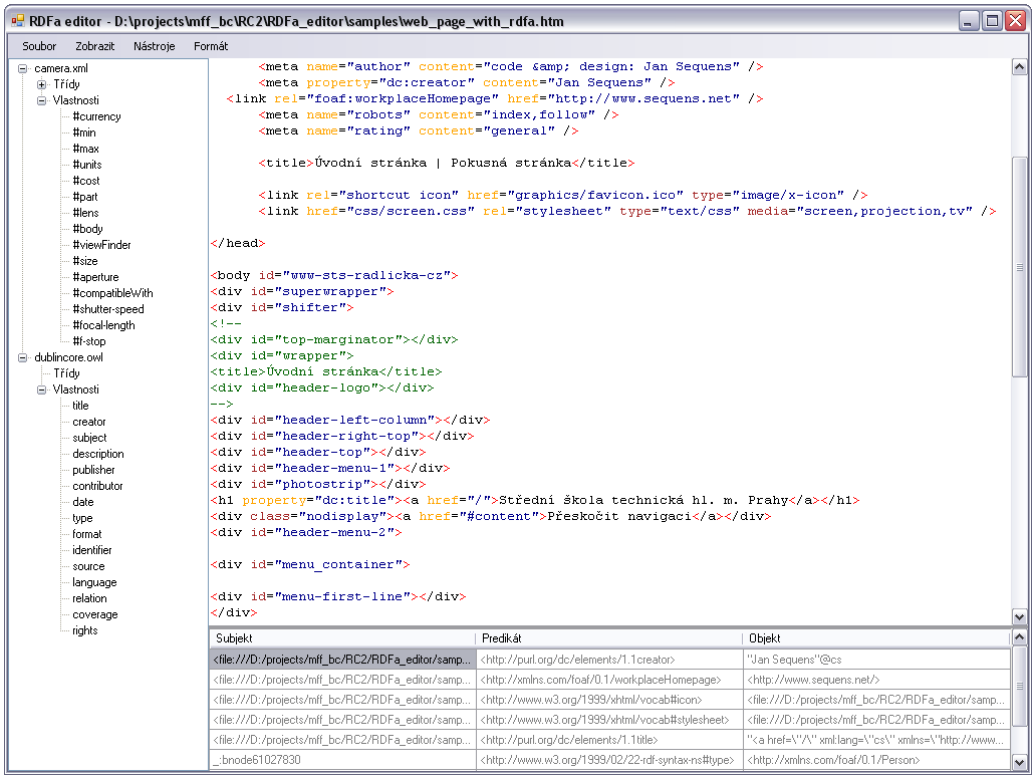
Program byl vyvíjen a testován na operačním systému Microsoft Windows XP. Teoreticky by měl být použitelný na jakémkoliv systému, který je kompatibilní s platformou Microsoft .NET verze 2.0 či vyšší.

Instalaci programu provedeme spuštěním souboru *setup.exe* v kořenovém adresáři. Pokud na cílovém počítači není instalován .NET verze alespoň 2.0, instalátor se jej pouze pokusí automaticky stáhnout a nainstalovat. Poté program spustí.

Pokud je cílový počítač vybaven platformou .NET verze alespoň 2.0, je možné program pouze spustit souborem *RDFa_editor.exe* v adresáři *RDFa_editor*.

4.3 Základy práce s programem

Jak je patrné z obrázku 3.1, program je tvořen jediným oknem, rozděleným na několik částí.



Obrázek 4.1: Ukázka prostředí programu.

V levé části je prostor pro zobrazení importovaných ontologií, v dolní části se mohou zobrazovat získané RDF trojice a zbylý prostor je vyhrazen editaci textu.

4.3.1 Vytvoření nového souboru

Nový soubor vytvoříme volbou *Soubor – Nový* v hlavním menu nebo klávesovou zkratkou *Ctrl + N*.

4.3.2 Zavření souboru

Soubor zavřeme následující volbou v hlavním menu: *Soubor – Zavřít*.

4.3.3 Otevření souboru

V menu *Formát – Znaková sada pro otevírání souborů* vybereme znakovou sadu, ve které je kódován soubor, který chceme otevřít. Poté otevřeme soubor volbou *Soubor – Otevřít...* v hlavním menu nebo klávesovou zkratkou *Ctrl + O*.

4.3.4 Uložení souboru

V menu *Formát – Znaková sada pro ukládání souborů* vybereme znakovou sadu, ve které chceme ukládaný soubor kódovat, případně ponecháme volbu *Stejná jako pro otevírání* (doporučeno). Poté soubor uložíme volbou *Soubor – Uložit* nebo *Soubor – Uložit jako...* v hlavním menu. Pro prosté uložení můžeme použít klávesovou zkratku *Ctrl + S*.

4.3.5 Zapnutí a vypnutí zvýrazňování syntaxe

Program umožňuje zvýrazňovat XHTML syntaxi. Zapnutí či vypnutí této funkce provedeme volbou *Zobrazit – Zvýrazňovat syntaxi* v hlavním menu nebo klávesovou zkratkou *Alt + S*.

4.3.6 Vložení deklarace typu dokumentu

Pro snadné vložení deklarace typu dokumentu pro XHTML dokument s RDFa anotací na pozici kurzoru použijeme volbu *Nástroje – Vložit na pozici kurzoru XHTML+RDFa doctype* nebo klávesovou zkratku *Ctrl + D*.

4.4 Zobrazování extrahovaných RDF trojic

4.4.1 Zapnutí a vypnutí zobrazení RDF trojic

Program umožňuje zobrazit RDF trojice extrahované z RDFa anotace XHTML dokumentu. Zapnutí či vypnutí této funkce provedeme volbou *Zobrazit – RDF trojice* v hlavním menu nebo klávesovou zkratkou *Alt + R*. Trojice se zobrazují v dolní části programu. Velikost oblasti pro zobrazení trojic lze nastavit tažením za horní okraj této oblasti.

4.4.2 Aktualizace RDF trojic

Seznam RDF trojic se z důvodů výpočetní náročnosti neaktualizuje průběžně. Možnou neaktuálnost trojic indikuje jejich zešednutí. Pokud chceme trojice aktualizovat, učiníme tak volbou *Nástroje – Aktualizovat RDF trojice*, klávesovou zkratkou *Ctrl + K* nebo kliknutím pravým tlačítkem do oblasti zobrazení trojic vyvoláme kontextovou nabídku a zvolíme *Aktualizovat*.

Část programu, která zajišťuje extrakci RDF trojic, předpokládá korektní XHTML+RDFa soubor. Pokud je po aktualizaci trojic seznam nečekaně prázdný, nachází se dokument v nekorektním stavu (chybějící koncové uvozovky u hodnoty parametru apod.).

4.5 Práce s nápovědou atributů

Program nabízí možnost automatické kontextové nápovědy RDFa atributů během psaní, která urychluje jejich zadávání.

4.5.1 Zapnutí a vypnutí nápovědy atributů

Zapnutí či vypnutí této funkce provedeme volbou *Zobrazit – Napovídat atributy* v hlavním menu nebo klávesovou zkratkou *Alt + N*.

4.5.2 Vyvolání nápovědy atributů

Vyvolání kontextové nápovědy se děje automaticky během zadávání textu. Ovlivňuje ho pozice kurzoru v textu a kontext, ve kterém se kurzor nachází. Je zřejmé, že je snahou, aby se nápověda objevovala „v pravý čas na pravém místě“.[1] Požádat o nápovědu lze také explicitně stiskem klávesové zkratky *Ctrl + mezerník*.

4.6 Práce s ontologiemi

4.6.1 Přidání ontologie

Program umožňuje importovat pojmy z OWL ontologií pro usnadnění další práce. Ontologie přidáme volbou *Soubor – Přidat ontologii...* v hlavním

[1] Uvedení přesného algoritmu by bylo na tomto místě zbytečné a zdlouhavé.

menu nebo klávesovou zkratkou *Ctrl + T*. Po vybrání souboru obsahujícího definici ontologie se v případě úspěšného importu ontologie objeví v levé části programového okna.

4.6.2 Použití pojmů z přidané ontologie

Úspěšně přidaná ontologie se zobrazí jako položka v levé části programového okna, reprezentovaná názvem souboru, ve kterém byla ontologie uložena. Symbol znaménka plus u položek v této časti indikuje, že je položka kliknutím na zmíněný symbol „rozbalitelná“.

Rozbalením přidané ontologie a jejích dvou potomků zobrazíme OWL třídy a vlastnosti získané při importu ontologie. Toto zobrazení nám může pomoci pro lepší orientaci v ontologii.

Tyto pojmy můžeme však také vkládat do textu. To provedeme přetažením („drag&drop“) příslušného pojmu do textu. Na místo, kde pojem v textu „upustíme“, se vloží

- kompaktní URI (CURIE) pojmu, je-li v elementu `html` definován prefix pro část URI vkládaného pojmu, [2]
- plné URI pojmu v opačném případě.

4.6.3 Odebrání ontologie

Pro odebrání dříve přidané ontologie vyvoláme nad ontologií pravým tlačítkem kontextovou nabídku a zvolíme *Odstranit*.

[2] Dokonalejší by jistě bylo hledat definici prefixu ve všech elementech, které jsou nadřazeny elementu, do kterého je pojem vkládán. To se ovšem zatím nepodařilo implementovat kvůli přílišné časové náročnosti takového hledání.

Kapitola 5

Implementace

5.1 Platforma

Jako platformu pro tvorbu RDFa editoru jsem zvolil Microsoft .NET. Hlavním důvodem byla moje pozitivní zkušenost s touto technologií. Program je psán v jazyku C# za použití vývojového prostředí Microsoft Visual Studio 2005.

5.2 Struktura programu

Program je rozčleněn do několika tříd, které spolu vzájemně spolupracují:

- `DebugForm` – formulář pro výpis hlášení v ladícím režimu,
- `MainForm` – hlavní formulář aplikace obsahující základní funkčnost,
- `OwlOntology` – třída pro reprezentaci zpracované ontologie,
- `OwlParser` – základní třída pro zpracování ontologií,
- `OwlTriplesProcessor` – třída pro zpracování ontologií,
- `RdfaTriplesProcessor` – třída pro zpracování RDF trojic,
- `Settings` – třída pro uchování nastavení programu,
- `SQRichTextBox` - komponenta rozšiřující standardní komponentu `System.Windows.Forms.RichTextBox`,

- `SyntaxHelper` – třída pro automatické napovídání atributů,
- `SyntaxHighlight2` – třída pro zvýrazňování syntaxe.

Kromě těchto tříd program využívá také následující prvky třetích stran:

- C# knihovnu *SharkBox* pro parsing RDFa jazyka, viz 5.3.1,
- C# knihovnu *SemWeb* pro práci s RDF trojicemi, viz 5.3.2.

5.3 Využití prvků třetích stran

5.3.1 Knihovna SharkBox

SharBox je knihovna napsaná v jazyce C#, za jejíž tvorbou stojí „Institut für Datenbanken und Informationsysteme, Universität Freiburg“ [1].

Tato knihovna je určena k parsingu RDFa. Za pomoci knihovny SemWeb (viz 5.3.2) získává z RDFa zápisu RDF trojice.

Knihovna je volně šiřitelná pro akademické a vzdělávací užití [1].

5.3.2 Knihovna SemWeb

SemWeb, plným jménem „Semantic Web/RDF library“, je open-source C# knihovna po čtení, zápis a serializaci RDF trojic. Mezi její další schopnosti patří také podpora dotazování jazykem SPARQL.

Jejím autorem je Joshua Tauberer a je šířena v licenci GNU GPL [4].

5.4 Práce se soubory

5.4.1 Otevírání souborů

Program umožňuje otevírání souborů ve třech různých kódováních. Konkrétní kódování vybírá uživatel v menu programu před otevřením souboru.

Načtení souboru v zadaném kódování obstarává kód, jehož klíčovou část vidíme zde:

```
FileStream stream = new FileStream(
                    mmOpenFileDialog.FileName, FileMode.Open);
string final = String.Empty;
byte[] buffer = new byte[Settings.BUFSIZE];

using (MemoryStream ms = new MemoryStream())
{
  while (true)
  {
    int read = stream.Read(buffer, 0, buffer.Length);
    if (read <= 0) // konec filestreamu
    {
      final = Settings.openFilesIn.GetString(ms.ToArray());
      break;
    }
    ms.Write(buffer, 0, read);
  }
  ms.Close();
}
```

Pomocí vytvořeného souborového proudu typu `FileStream` načítáme vždy část souboru do proměnné `buffer` (s pevně danou délkou) a obsah této proměnné přidáváme do paměťového proudu typu `MemoryStream`.

Po dosažení konce souborového proudu převedeme funkcí `ToArray()` obsah paměťového proudu na pole bajtů a předáme ho funkci `GetString()`, zavolané nad uživatelem zvoleným kódováním.

5.4.2 Ukládání souborů

Ukládání souborů je rovněž umožněno ve třech různých kódováních. Uveďme si opět fragment kódu, který je za ukládání souborů zodpovědný.

```
FileStream fs = new FileStream(fileUri, FileMode.Create);
BinaryWriter bw = new BinaryWriter(fs);
Encoding enc = Settings.saveAsOpened ?
  Settings.openFilesIn : Settings.saveFilesIn;
bw.Write(enc.GetBytes(codeBox.Text));
bw.Close();
```

```
fs.Close();
```

K zápisu používáme souborový proud typu `FileStream` a proud `BinaryWriter`. Pomocí proměnné `enc` typu `System.Text.Encoding` získáme již kódovaný text jako pole bajtů, které předáme metodě `Write()` binárního proudu.

5.5 Zvýrazňování syntaxe

5.5.1 Idea

Zvýrazňování syntaxe je funkcí, se kterou se běžně setkáváme v programátorských editorech a vývojových prostředích. Rozumí se jí obarvování částí textu podle jejich významu v daném programovacím nebo značkovacím jazyce, což významně zvyšuje přehlednost zápisu. Obarvení má být pouze vizuálním efektem a v žádném případě nemá jakkoliv měnit či ovlivňovat obsah souborů, které zvýrazňovaný kód obsahují.

Funkce zvýrazňování syntaxe – taková, jakou ji známe z programátorských editorů – se skládá ze dvou základních částí:

- samotná analýza (parsing) textu spojená s jeho obarvením a
- určování analyzované oblasti textu a zajištění obnovy jejího obarvení.

Obě výše uvedené části se musí provádět pokud možno takovým způsobem a v takové okamžiky, aby uživatel vždy viděl korektně obarvený text.

V RDFa editoru je funkce zvýrazňování syntaxe reprezentována třídou `SyntaxHighlight2`. Umí zvýrazňovat syntaxi XHTML+RDFa dokumentů, obecně také jakéhokoliv XML dokumentu.

5.5.2 Způsoby analýzy textu

Text lze procházet a analyzovat několika způsoby. Při tvorbě programu jsem se rozhodoval mezi dvěma takovými z nich.

Prvním z nich bylo použít speciální třídu `System.Xml.XmlTextReader` pro čtení jazyka XML, která je standardní součástí .NET. Ta umožňuji pohodlné zpracování XML zhruba tímto způsobem:

```
while (reader.Read())
{
  switch (reader.NodeType)
    {
      case XmlNodeType.Element: // prvek
            // ... tisk obarveneho retezce
            break;
      case XmlNodeType.Attribute: // atribut
            // ... tisk obarveneho retezce
            break;
      // ...
}
```

Jak se však ukázalo, zásadním nedostatkem tohoto přístupu je nedostatečná rychlost. Načtení a vytištění textu naznačeným způsobem o délce přibližně jednoho kilobajtu trvalo mnoho sekund. Tento pohodlný způsob se tedy bohužel ukázal jako nepoužitelný.

Druhým způsobem, který jsem nakonec zvolil, bylo čtení textu po znacích a jeho zpracování vlastními silami.

Metodou, která toto zpracování provádí, je privátní instanční metoda `void parseText(int offset, int stopPos, RichTextBox rtb)`. Jejími parametry je proměnný typu `System.Windows.Forms.RichTextBox` a dvě celá čísla, která vyjadřují počáteční a koncový index znaku, mezi kterými má být text analyzován.

Základem této metody je `for` cyklus procházející všechny znaky v daném rozsahu a několik proměnných typu `bool` pro uložení příznaků. Pokud je zpracovávaný znak otevírací lomenou závorkou, začíná nový XML element, ukončovací lomená závorka indikuje naopak opuštění elementu.

Je-li nastaven příznak, že se nacházíme uvnitř XML elementu, nastavujeme sledováním uvozovek a bílých znaků příznaky týkající se atributů a jejich hodnot. Tyto příznaky průběžně vyhodnocujeme a podřizujeme jim barvení textu.

5.5.3 Způsoby barvení textu

Jedinou standradní .NET komponentou, která umožňuje obarvování textu, je `System.Windows.Forms.RichTextBox`. Rozhodl jsem se ji tedy využít. K

samotnému obarvování textu pak lze přistoupit jedním ze dvou možných způsobů.

Prvním z nich je využití vlastnosti `SelectionColor`[1], kterou disponují instance třídy `RichTextBox`.

Následující fragment kódu ukazuje princip zabarvení části textu pomocí této vlastnosti:

```
System.Windows.Forms.RichTextBox rtb =
  new System.Windows.Forms.RichTextBox();
rtb.Text = "Semanticky web ma velkou budoucnost.";
rtb.SelectionStart = 11;
rtb.SelectionLength = 3;
rtb.SelectionColor = System.Drawing.Color.Red;
```

Tento kód v komponentně `rtb` typu `RichTextBox` nejprve označí text od znaku na pozici 11 do znaku na pozici 13 (délka označení 3) a tomuto textu přiřadí červenou barvu.

Toto řešení je funkční, nicméně jsem na něj pohlížel jako na řešení ne zcela elegantní a přímé.

Druhým přístupem k barvení textu uvnitř komponenty `RichTextBox` je úprava surových RTF[2] dat, které jsou k dispozici skrze vlastnost `Rtf`. RTF data jsou ukládána jako ASCII znaky, lze s nimi tedy pracovat jako s obyčejným řetězcem.

Principem tohoto řešení je vložit do RTF hlavičky připravenou tabulku barev (*color table*, což je řetězec s definovanou strukturou, a poté do RTF dat vkládat řídící sekvence, které přepínají barvu textu, jenž za nimi následuje.

Příklad tabulky barev v RTF:

```
{\colortbl ;\red255\green0\blue0;\red192\green192\blue192;}
```

[1]Zde rozumíme „vlastností“ konstrukt jazyka C# označovaný jako *property*

[2]Rich Text Format; volně šiřitelný formát pro uložení formátovaného textu od společnosti Microsoft

Tato tabulka definuje dvě barvy – jasně červenou (s indexem 1) a světle šedou (s indexem 2). Index 0 je vždy implicitně přiřazen výchozí barvě komponenty. Změny barev v textu provádíme způsobem naznačeným zde:

```
\cf1 toto je cerveny text\cf0 toto je normalni text
```

Na začátku vývoje jsem se rozhodl pro práci se surovým RTF, neboť jsem to považoval za čistší a pravděpodobně časově efektivnější řešení. Od toho jsem posléze ustoupil z důvodů popsaných v 5.5.4 a přiklonil se k metodě s vlastností `SelectionColor`. Zdrojové kódy RDFa editoru, které jsou součástí této práce, pro zajímavost obsahují soubor s třídou `SyntaxHighlight1`, ve které bylo výše popsané barvení přímou úpravou RTF implementováno. Program tuto třídu ale nepoužívá.

5.5.4 Aktualizace obarvení

Umíme-li text správně obarvit, musíme také zajistit, že se jeho obarvení „přepočítá“ pokaždé, když to bude třeba.

Nejjednodušší možnost je přepočítat obarvení celého textu po každé jeho změně (událost `TextChanged`). Tím je zajištěno, že text bude vždy správně obarven. Nepřekonatelnou překážkou tohoto řešení je ale neúměrná časová náročnost, která se výrazně projeví již při barvení souborů o délce několika kilobajtů.

Druhou možností je pak obarvování pouze té části textu, kterou uživatel právě vidí na obrazovce. To teoreticky přinese značné urychlení, které bude tím významnější, čím bude zpracovávaný soubor delší. Nevýhodou je možné nesprávné obarvení začátku viditelné části textu např. v případě, že XML element začíná na řádce, která ještě viditelná není.

Z důvodů uvedených v 5.5.8 jsem potřeboval barvený text neupravovat přímo v používané `RichTextBox` komponentně, ale zkopírovat si ho na jiné místo a tam ho upravovat. K tomuto účelu se nabízela vlastnost `SelectedRtf`.

Pomocí vlastností `SelectionStart` a `SelectionLength` jsem metodou uvedenou v 5.5.5 získal číslo první viditelné řádky, označil viditelnou část textu a vlastností `SelectedRtf` si zkopíroval jeho RTF kód.

V pomocné komponentě `RichTextBox` jsem text obarvil a za opětovného použití `SelectedRtf`zaměnil za text původní.

Vše fungovalo tak, jak mělo, ale jen na první pohled. Při vkládání RTF pomocí přiřazení do vlastnosti `SelectedRtf` totiž dochází k mnou doposud nepochopeným vedlejším jevům. Text se zobrazí tak, jak má, ale komponenta posléze místy vykazuje nepochopitelné chování. Na jejím začátku se objevují nežádoucí znaky mezer, které

- způsobují nesmyslné chování kurzoru při uživatelských úpravách textu v komponentně (přeskakování apod.),
- nejdou programově smazat (nepomáhá nastavení vlastnosti `Text` ani `Rtf` na `String.Empty`, nepomáhá volání metody `Clear()`).

Toto chování jsem ani po dlouhém zkoumání a hledání informací nedokázal vysvětlit. Z tohoto důvodu jsem se rozhodl nepoužívat vlastnost `SelectedRtf`, ale výlučně pouze `Rtf`. Proto se v programu do pomocné `RichTextBox` komponenty kopíruje vždy RTF příslušící textu celého souboru.

Obarvení je však přepočítáno pouze pro tu část textu, která je pro uživatele viditelná. Z toho důvodu dochází k aktualizace obarvení nejen při události `TextChanged`, ale také při události `ScrollingFinished`, která byla přidána do komponenty `SQRichTextBox` (viz 5.5.5).

5.5.5 Obnovení pozice kurzoru

Každé nastavení vlastnosti `Rtf` způsobí, že se kurzor i pozice scrollbaru[3] nastaví zcela na začátek. V případě zvýrazňování syntaxe je ale nutné, aby si uživatel nastavení vlastnosti `Rtf` ideálně vůbec nevšiml.

Docílíme toho zapamatováním si pozice kurzoru v textu a čísla první viditelné řádky v komponentě před vložením nového RTF. Ihned po vložení tyto údaje opět nastavíme na původní hodnoty.

Narážíme však na poměrně nepříjemný problém – .NET nemá implementovány vhodné metody na práci s pozicí scrollbaru. Nejen tato skutečnost způsobila, že jsem se rozhodl pro vytvoření vlastní komponenty odvozené od `System.Windows.Forms.RichTextBox`. Pojmenoval jsem ji `SQRichTextBox`.

První přidanou vlastností je `FirstVisibleLine`. Uveďme si opět krátký fragment kódu:

[3] český ekvivalent je „rolovací lišta“ nebo „posuvník“; v této práci se budu držet originálního anglického termínu

```
[DllImport("user32.dll")]
static extern int SendMessage(
  IntPtr hWnd, int wMsg, int wParam, int lParam);

public int FirstVisibleLine
{
  get
  {
    return SendMessage(this.Handle, GET_FIRST_VISIBLE_LINE, 0, 0);
  }
  set
  {
    SendMessage(this.Handle, WM_VSCROLL, SB_TOP, 0);
    SendMessage(this.Handle, EM_LINESCROLL, 0, value);
  }
}
```

Deklarovali jsme externí statickou metodu `SendMessage()`, která je implementována v knihovně `user32.dll`. To jsme uvedli použitím atributu `DllImport`. Knihovna `user32.dll` je součástí operačního systému Windows, jedná se o neřízenou (*unmanaged*) DLL[4] knihovnu – její kód tedy není interpretován CLR[5].

Funkce `SendMessage()`zasílá specifikovanou zprávu[6] určenému oknu. Pro předání dodatečných informací ke zprávě slouží poslední dva parametry.

Jména zpráv jsou konstanty definované v hlavičkových souborech systému Windows, pro jejich použití v C# je třeba je „ručně“ definovat.

V metodě `get` v našem příkladu žádáme zprávou `GET_FIRST_VISIBLE_LINE` o vrácení první viditelné řádky.

V metodě `set` nejdříve první zprávou posuneme scrollbar zcela nahoru, druhou zprávou pak provedeme posunutí scrollbaru o daný počet řádek níže.

[4]Dynamic-Link Library – dynamicky připojovaná knihovna

[5]Common Language Runtime; virtuální stroj platformy .NET

[6]zprávy systému Microsoft Windows; základní prostředek pro komunikaci mezi jednotlivými okny

5.5.6 Problém „problikávání“

Jelikož každé přepočítání obarvení textu zahrnuje označování textu, posunování scrollbaru atd., komponenta při tom nehezky problikává.

Tento problém jsem vyřešil dvojím voláním metody `LockWindowUpdate()` z knihovny `user32.dll` (viz 5.5.5), jak je naznačenovo v této ukázce:

```
LockWindowUpdate(codeBox.Handle);
// ... práce s komponentou codeBox
LockWindowUpdate(IntPtr.Zero);
```

První volání této metody zakáže překreslování komponenty `codeBox` (typu `SQRichTextBox`), druhé ho opět povolí.

5.5.7 Problém hladkého skrolování

Komponenta `RichTextBox` má implementováno tzv. „hladké“ skrolování (*smooth scrolling*). Tímto pojmem se rozumí způsob skrolování, při kterém je obsah okna posunován po jednotlivých obrazových bodech, nikoliv po celých řádkách textu. Pro hladké skrolování je také typický určitý „dojezd“ neboli plynulé a postupné zastavení posouvaného obsahu při ukončení skrolování.

Toto chování přináší v některých případech vyšší uživatelský komfort, ale v případě programátorských editorů není žádoucí ani obvyklé. Dále pak, jak je vysvětleno v 5.5.5, obnova polohy scrollbaru je v tomto programu zajištěna s přesností výšky jedné řádky. Rádi bychom tedy hladké skrolování v naší komponentě zakázali.

Bohužel, .NET nenabízí vypnutí tohoto chování v žádné vlastnosti komponenty, jak by někdo mohl předpokládat. Pro naplnění tohoto požadavku vlastními silami je potřeba zpracovávat zprávu `WM_VSCROLL`, kterou zasílá scrollbar svému rodičovskému oknu.

To provedem překrytím metody `WndProc()` v komponentě `SQRichTextBox` a odchytáváním zmíněné zpávy. V tomto programu je ovšem naplněn požadavek zrušení hladkého skrolování za cenu toho, že se při pohybu táhlem scrollbaru průběžně nezobrazuje obsah okna dle polohy táhla. Zpráva je totiž v případě, že je v dolním slově parametru `wParam` hodnota `SB_THUMBTRACK`, indikující tažení, blokována. Obsah okna je změněn až poté, co uživatel táhlo pustí, což je indikováno hodnotou `SB_POSITION` v dolním slově paramateru `wParam`.

V této chvíli je provedeno „zaokrouhlení pozice na výšku řádku“ tak, aby byl první viditelný řádek textu viditelný zcela a ne jen zčásti:

```
int rem = hiWord % line_height;
if (rem != 0)
{
  hiWord += (rem >= line_height / 2) ?
    (line_height - rem) : (-rem);
  int newWParam = (loWord & 0xFFFFF) | (hiWord << 16);
  SendMessage(msg.HWnd, msg.Msg, newWParam, 0);
  // ...
  return;
}
```

Určíme, o kolik pixelů je pozice (obsažená v horním slově parametru `wParam`) dále, než bychom chtěli. Podle této hodnoty poté rozhodneme, zaokrouhlíme-li pozici nahoru či dolů, zkonstruujeme nový `wParam` a pošleme novou, „opravnou“ zprávu. Původní zprávu nedáme ke zpracování překrývané metodě.

Podobným způsobem odchytáváme také zprávu `WM_MOUSEWHEEL` indikující točení kolečkem myši.

5.5.8 Asynchronní spouštění

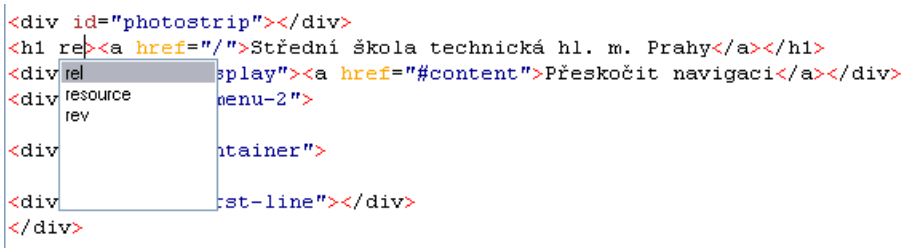
Spouštíme-li zvýrazňování syntaxe synchronně (tedy standardním způsobem), program vždy čeká na dokončení barvení textu. Během jeho provádění nepřijímá vstupy uživatele, nezpracovává zprávy operačního systému a celkově se nachází ve zdánlivě „zamrzlém“ stavu, což jistě není nic hezkého ani praktického.

Abychom tento problém vyřešili, je nutné zvýrazňování syntaxe spouštět asynchronně neboli ve zvláštním vlákně, což umožní pseudoparalelní[7] zpracování obou úkolů – zvýrazňování syntaxe i příjem vstupů uživatele apod.

Abychom mohli zvýrazňování syntaxe spouštět ve zvláštním vlákně, je třeba provádět barvení textu jinde než přímo v zobrazované komponentě.

Po otevření souboru, při změně textu a po změně pozice scrollbaru je spouštěna metoda `doSyntaxHighlight()`. Ta v případě, že ještě běží minulé vlákno zvýrazňování syntaxe, toto vlákno zastaví. Poté vytvoří a

[7] v případě vícejaderného procesoru teoreticky i skutečně paralelní



Obrázek 5.1: Okénko nápovědy atributů.

spustí vlákno nové. Tím je zajištěno, že se do `SQRichTextBox` komponenty nezkopíruje korektně obarvený, leč neaktuální text.

Metoda `daemonEdit()`, která je vláknem spouštěna, na svém začátku předem daný interval (daný ve třídě `Settings`, např. 300 milisekund) čeká. Důvodem je, aby program vyčkal se skutečným zvýrazňováním syntaxe na okamžik, kdy uživatel dokončí rychlejší sled stisků kláves (např. psaní jedno slova) a nezatěžoval mezitím zbytečně procesor. Každé spuštěné vlákno bude totiž v takovém případě zrušeno dřív, než začně skutečně pracovat.

5.6 Napovídání atributů

Základ funkce napovídání atributů je implementován ve třídě `SyntaxHelper`, na jeho práci se však podílí i jiné části programu. Účelem této funkce je urychlení psaní názvů předdefinovaných XML atributů, což poskytne uživateli větší komfort.

5.6.1 Princip automatického napovídání atributů

Základem nápovědy je komponenta `System.Windows.Forms.ListBox` (říkejme jí „okénko nápovědy“), ve které se zobrazuje seznam dostupných XM atributů, které je možno jednoduše vložit. Seznam nabízených atributů se mění podle toho, jaké znaky uživatel zadává – program se snaží doplnit to, co uživatel začal psát.

Okénko nápovědy se nezobrazuje stále, ale jen pokud uživatel začíná psát či již píše název XML atributu. Zobrazuje se pod kurzorem, případně nad ním.

Základní metodou, která je volána z ostatních tříd, je instanční metoda `whisper()`, která přizpůsobí okénko nápovědy vzniklé situaci.

5.6.2 Nalezení kontextu kurzoru

Aby se mohlo rozhodnout o zobrazení okénka nápovědy, je potřeba znát „kontext kurzoru“, kterým je míněna znalost, v jaké části XML se kurzor nalézá (uvnitř elementu, uvnitř atributu, uvnitř hodnoty atributu). To zajišťuje metoda `findContext()`, která určí hodnoty několika příznaků prohledáním textu směrem zpět před kurzor. Prohledávání je ukončeno kteroukoliv z těchto tří událostí:

- narazili jsme na levou či pravou lomenou závorku,
- narazili jsme na začátek souboru,
- vyčerpali jsme maximální počet prohledávaných znaků, definovaný ve třídě `Settings`.

V určitých případech není třeba pro rozhodnutí o nezobrazení okénka nápovědy znovu volat metodu `findContext()`. Jedná se o situaci, kdy již není žádný vhodný atribut k nápovědě a uživatel pouze napíše další nebílý znak. Tím situaci jistě „nezlepší“ a stále nebude žádný atribut k dispozici.

5.6.3 Blokované klávesy

Pokud je zobrazeno okénko nápovědy, je potřeba, aby některé klávesy získaly speciální význam – sloužily pro ovládání okénka nápovědy a neměly vliv na komponentu s XML textem. Jedná se například o šipky nahoru a dolů pro pohyb mezi nabízenými atributy, klávesu Enter pro potvrzení vložení nabízené položky či klávesu Escape pro zrušení nápovědy.

K tomuto důvodu má komponenta `SQRichTextBox`vlastnost `BlockedKeys`, pomocí které lze přidávat a odebírat klávesy či klávesové kombinace, které nebudou interpretovány, ale při jejich stisku bude vyvolána speciální událost `KeyDownBlocked`.

5.7 Zobrazování RDF trojic

Zobrazování RDF trojic získaných z RDFa atributů XHTML dokumentu je prováděno s využitím knihovny SharkBox (viz 5.3.1), která využívá knihovny SemWeb (viz 5.3.2). Práce s těmito knihovnami se odehrává v metodě `updateTriples()` v hlavním formuláři a využívá třídu `RdfaTriplesProcessor`.

5.7.1 Zpracování RDFa pomocí knihoven

„Tovární“ metodou `Shark.SharkParser.CreateParser()` vytvoříme novou instanci třídy `Shark.SharkParser`. Prvním parametrem této metody určujeme způsob zpracování zdrojového XML, v našem případě vybíráme paměťově náročnější, ale jistě spolehlivý [1] způsob DOM[8]. Druhý parametr je typu `SemWeb.MemoryStore`a představuje úložiště pro získané RDF trojice.

Jelikož výkonná metoda `Parse()`, kterou voláme nad instancí třídy `Shark.SharkParser`, požaduje jako svůj jediný parametr URI, uložíme text z komponenty `SQRichTextBox` do dočasného souboru, který této metodě pomocí jeho URI předáme. Dočasný soubor poté samozřejmě smažeme.

5.7.2 Další zpracování RDF trojic

Po uložení RDF trojic do proměnné typu `SemWeb.MemoryStore` zavoláme nad touto proměnnou metodu `Select()`, jejíž parametr je instance třídy `RdfaTriplesProcessor` implementující rozhraní `SemWeb.StatementSink`.

Tato třída RDF trojice uloží do proměnné typu `System.Data.DataSet`, na kterou posléze navážeme samotnou komponentu pro zobrazování RDF trojic.

5.8 Přidávání ontologií

Přidávání ontologií je prováděno s využití tříd `OwlParser`, `OwlOntology`, `OwlTriplesProcessor`a knihovny SemWeb (viz 5.3.2).

5.8.1 Zpracování přidávané ontologie

Ve instanci třídy `OwlParser`pomocí několika tříd a funkcí knihovny SemWeb načteme vybraný soubor s popisem ontologie jako RDF.

V instanci třídy `OwlTriplesProcessor`, která implementuje rozhraní `SemWeb.StatementSink`, vytřídíme URI, která reprezentují OWL třídy a vlastnosti. Ty poté pomocí instance třídy `OwlOntology` předáme třídě hlavního formuláře.

[8] Document Object Model – objektový model dokumentu

5.8.2 Přidání ontologie do stromu ontologií

Samotné přidání ontologie do stromu ontologií je pak zajištěno metodou `addOntologyToTree()`, náležící hlavnímu formuláři.

5.8.3 Přetahování pojmů z ontologií

Program umožňuje přetahování pojmů z ontologií metodou „drag&drop“. To je zajištěno pomocí obsluhy události `MouseDown` ve stromu ontologií a obsluhou událostí `DragEnter`, `DragOver` a `DragDrop` v komponentě `SQRichTextBox`.

Kapitola 6

Závěr

V závěru této práce bych se rád ještě jednou zamyslel nad otázkou volby platformy a krátce shrnul splnění cíle práce.

6.1 Zhodnocení volby platformy

Programování nad platformou .NET v jazyku C# je většinu času „příjemným zážitkem“. Jelikož Microsoft Visual Studio poskytuje možnost grafického návrhu aplikace, je možné soustředit se výhradně na logiku programu. Velmi příjemné jsou také možnosti ladění programu a systém IntelliSense pro kontextovou nápovědu.

Pokud je však vyžadována funkcionalita, kterou .NET neimplementuje, je její dosažení často vykoupeno určitou chybějící elegancí způsobu jejího řešení.

Nepříjemná situace také může nastat při používání některých vlastností komponent, neboť není zřejmé, co přesně metody pro čtení a zápis těchto vlastností provádějí.

6.2 Splnění cíle práce a možná rozšíření

Cíl práce – vytvoření editoru usnadňujícího RDFa anotaci XHTML dokumentů – byl podle mého názoru splněn.

Jsem si však vědom toho, že editor trpí určitými nedostatky v jeho ovládání, jejichž vyřešení by jistě přispělo k jeho větší použitelnosti.

Prostor pro další možná rozšíření této práce vidím zejména ve složitější analýze importovaných ontologií, která by byla využita pro důmyslné automatické napovídání tříd a vlastností z těchto ontologií a pro kontrolu korektní anotace dokumentu v souladu s definicí ontologie.

V neposlední řadě jsem si touto prací podstatně rozšířil obzor znalostí v oblasti sémantického webu a získal cenné zkušenosti s programováním na platformě .NET.

Literatura

[1] *Home of SharkBox*, http://shark.informatik.uni-freiburg.de

[2] *W3C – RDF Primer*,
http://www.w3.org/TR/2004/REC-rdf-primer-20040210

[3] *W3C – RDFa in XHTML: Syntax and Processing*,
http://www.w3.org/TR/rdfa-syntax

[4] *Semantic Web/RDF Library for C#/.NET*,
http://razor.occams.info/code/semweb

[5] *TopQuadrant – TopBraid Composer*,
http://www.topquadrant.com/topbraid/composer

[6] Gruber T.R.: *A Translation Approach to Portable Ontology Specifications*, Knowledge Acquisition, 5(2) (1993).

Dodatek A

Obsah přiloženého CD

- `/dotnetfx` – složka s instalačními soubory .NET
- `/RDFa editor_1_2_0_1` – složka programu RDFa editor; v případě počítače s již nainstalovaným .NET je možné program spouštět přímo odtud
- `/sample_files` – ukázkové soubory pro vyzkoušení práce s programem
- `/source_code` – zdrojové kódy programu
- `autorun.inf` – pomocný soubor (spouštění CD)
- `RDFa editor.application` – pomocný soubor
- `RDFa editor_1_2_0_1.application` – pomocný soubor
- `rdfa_editor.pdf` – text této práce
- `setup.exe` – instalační soubor vytvořený nástrojem „Publisher“ v prostředí Visual Studio; v případě potřeby nainstaluje prostředí .NET